REMARKS

Claims 48 and 60 have been amended to correct a clerical error. Claims 1, 4-12, 31-39, 42-44, 46-51, 54-56, and 58-62 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 1, 4-12, 31-39, 42-44, 46-51, 54-56 and 58-62 under 35 U.S.C. § 103(a) as being unpatentable over "Error Handling Interface (H5E)" (hereinafter "H5E") in view of Lowen et al. (U.S. Patent 7,017,077) (hereinafter "Lowen"). Applicants respectfully traverse this rejection for at least the following reasons.

In regard to claim 1, the cited art does not teach or suggest *an error trace mechanism for a multithreaded program configured to: in each thread of the multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in a memory storage area specific to the thread in accordance with an application programming interface (API) to the error trace mechanism.* The H5E reference teaches an error handling interface in which errors that occur during execution of a library are pushed onto an error stack, and that the errors can be retrieved from the error stack. However, the H5E reference does not teach or suggest an error trace mechanism for a multithreaded program configured to, for each error generated by one or more functions executed in each of two or more threads in the multithreaded program, store an error trace element in a memory storage area specific to the corresponding thread, as is recited in claim 1. Further, the H5E reference does not teach or suggest an error trace mechanism configured to obtain an error trace for each of two or more threads of a multithreaded program, wherein each error trace includes one or more error trace elements specific to the corresponding thread, as is recited in claim 1.

The Examiner states that "H5E does not specifically disclose - multithreaded program extension of the above limitation." The Examiner goes on to assert that Lowen discloses "error retention for multithreaded programs". The Examiner asserts that Lowen, in col. 2, line 56 - col. 3, line 48 describes "how a logger can be instantiated for each thread", citing col. 3, lines 7-8, which states "The application thread may obtain an instance of the logger…". **However, a careful reading of Lowen reveals that Lowen does not teach or suggest that a new instance of a "logger" is constructed for *each* thread, as the Examiner asserts.** Instead, Lowen teaches that **one** (a *singleton*) instance of a logger is constructed, and all threads thereafter use that **one** instance. This is made clear in FIG. 3 and the accompanying discussion of the figure (col. 3, lines 24-32), which describes **how** "the application thread may obtain an instance of the logger":

> Illustrated in FIG. 3, is an embodiment of the logger as a Singleton design pattern object 300. When an instance of the logger is requested 310, the logger's static instance method may inquire a static variable indicating whether an instance of the logger has already been constructed 320. If the logger has not yet been constructed then the instance method will construct a new logger object 330. Upon exit of the instance method, the single instance of the logger object is returned 340.

From the above and from FIG. 3, it is clear that an instance of the logger is **only** constructed if the logger has not yet been constructed. If the logger has already been constructed (e.g., by a previous call to the static instance method), then the previously constructed instance is returned. Thus, all threads in Lowen would share a common **singleton** instance of Lowen's logger. That Lowen teaches a single instance of a logger is further illustrated elsewhere in the Lowen reference, for example in the Abstract:

> The method comprises executing an application which uses **a** logger that collects log statements, collecting at least one log statement from at least one application thread and storing the at least one log statement in memory.

In addition, FIG. 1 of Lowen illustrates one instance of logger 120 associated with Application 110. That the single instance of logger 120 is used by multiple threads is further expressed in the discussion of FIG. 1 at col. 2, lines 62-64:

> The logger 120 may respond to a request from threads of execution operating as part of a software application 110.

Lowen clearly does not teach that a separate instance of the logger 120 is constructed for each thread. Indeed, Lowen appears to teach <u>against</u> constructing an instance of logger 120 for each thread. Therefore, since Lowen requires the same logger for all threads, H5E in view of Lowen would not suggest in each thread of the multithreaded program to store an error trace element in a memory storage area <u>specific to the thread</u>, for each error generated by one or more functions executed in the thread, in accordance with an application programming interface (API) to the error trace mechanism.

The Examiner goes on to assert that Lowen teaches "errors generated by one or more functions executed in the thread can be stored on error trace element in a memory storage area specific to the thread". The Examiner cites col. 3, lines 46-48, "…by instantiating a log message queue…". Col. 3, lines 46-48 is included in a description of a "flowchart representation of a logger object's constructor 400 associated with the logger's instance method 300 of FIG. 3" (col. 3, lines 33-35). At col. 3, lines 28-30, Lowen states that "If the logger has not yet been constructed then the instance method will construct a new logger object 330." In other words, the method of FIG. 4 will only be executed <u>if the logger has not yet been constructed</u>. Since Lowen teaches that <u>only one instance</u> of the logger is constructed, the method of FIG. 4 will only be executed once (when "instance == NULL is true in FIG. 3), and therefore Lowen teaches that <u>only one instance of a log message queue will be generated</u>. Thus, Lowen does not suggest storing an error trace element in a memory storage area <u>specific to each thread</u>. **To the contrary, Lowen requires just the opposite.**

The Examiner goes on to assert that "Therefore, for each thread there is a corresponding logger with a corresponding instance of message queue to store error trace elements and the message queue is a memory storage area specific to the thread." **The Examiner has misinterpreted the reference.** As noted above, Lowen does **not** teach or suggest that "for each thread there is a corresponding logger". Instead, Lowen clearly teaches that there is <u>one</u> instance of a logger for all threads. Furthermore, Lowen does not teach or suggest that "for each thread there is a corresponding logger with a corresponding instance of message queue". Instead, Lowen clearly teaches that <u>only one instance of a logger is constructed</u>, and therefore <u>only one log message queue is generated</u>. Lowen's log message queue is clearly **not** "a memory storage area specific to the thread" as asserted by the Examiner.

Thus, the cited art, alone or in combination, does not teach the limitation *in each thread of the multithreaded program, for each error generated by one or more functions executed in the thread, <u>store an error trace element in a memory storage area **specific to the thread**</u>*, as is recited in claim 1 of the instant application.

Thus, for at least the reasons presented above, the rejection of claim 1 is not supported by the cited prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 1 also apply to claims 9, 31, 36, 39, 47, 51, and 59.

Furthermore, the Examiner has failed to state a *prima facie* rejection for claims 9, 31, 47, and 51. The Examiner only states in regard to claims 9, 31, 47, and 51 to "see reason of rejection of claim 1". However, claims 9, 31, 47, and 51 have different scope than claim 1. **Since the Examiner has failed to address the differences between the claims, the Examiner's rejection of claims 9, 31, 47, and 51 is additionally improper.**

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants respectfully submit that the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-69401/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: July 2, 2007